VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ

ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT

INSTITUTE OF INFORMATICS

NÁVRH DATBÁZE PRO ŘÍZENÍ UŽIVATELSKÝCH ÚČTŮ

DATABASE DESIGN FOR USER ACCOUNT MANAGEMENT

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — IVO KOSTRHOUN

AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ KŘÍŽ, Ph.D.

SUPERVISOR

BRNO 2010

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2009/10
Ústav informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Ivo Kostrhoun

Manažerská informatika (6209R021)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává bakalářskou práci s názvem:

Návrh datbáze pro řízení uživatelských účtů

v anglickém jazyce:

Database Design for User Account Management

Pokyny pro vypracování:

Úvod
Vymezení problému a cíle práce
Teoretická východiska práce
Analýza problému a současné situace
Vlastní návrhy řešení, přínos návrhů řešení
Závěr
Seznam použité literatury
Přílohy

Seznam odborné literatury:

DOSEDĚL, T. Počítačová bezpečnost a ochrana dat. 1. vydání. Brno: Computer Press, 2004. 190 s. ISBN 80-251-0106-1
HOTEK, M. Microsoft SQL Server 2008: krok za krokem. 1.vydání. Brno: Computer Press, 2009. 488 s. ISBN 978-80-251-2466-6
KOTLER, P. Marketing management. 1. vydání. Praha: Grada, 2007. 788 s. ISBN 978-80-247-1359-5
PUŽMANOVÁ, R. Moderní komunikační sítě od A do Z. 2. aktualizované vydání. Brno: Computer Press, 2006. 430 s. ISBN 80-251-1278-0
ŘEPA, V. Podnikové procesy. 2.rozšířené vydání. Praha: Grada, 2007. 281 s. ISBN 978-80-247-2252-8
VOŘÍŠEK, J. Principy a modely řízení podnikové informatiky. 1. vydání. Praha: Oeconomica, 2008. 446 s. ISBN 978-80-245-1440-6

Vedoucí bakalářské práce: Ing. Jiří Kříž, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2009/10.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ Fakulta podnikatelská
L.S.

Ing. Jiří Kříž, Ph.D.
Ředitel ústavu

doc. RNDr. Anna Putnová, Ph.D., MBA
Děkanka

V Brně, dne 7. 2. 2010

Abstrakt

Bakalářská práce se zabývá návrhem databáze pro efektivní řízení uživatelských účtů v systémech UNIX ve společnosti IBM. Databáze byla modelována v MS SQL 2008. Jejím úkolem bylo zlepšit práci administrátorů a zkvalitnit řešení problémů, týkající se řízení uživatelských účtů.

Abstract

This bachelor thesis deals with database design for effective user account management in UNIX systems in Brno IBM company. Database has been designed in MS SQL 2008. Its task was to improve administrators‘ work and to make better solution of problems dealing with user accounts management.

Klíčová slova

databáze, sql, informační systém,uživatelský, účet, řízení

Keywords

Database, sql, information system, user, account, management

Bibliografická citace práce:

KOSTRHOUN, I. *Návrh datbáze pro řízení uživatelských účtů.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2010. 49 s. Vedoucí bakalářské práce Ing. Jiří Kříž, Ph.D.

Čestné prohlášení

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 31. května 2010

……

Poděkování:
Rád bych poděkoval svému vedoucímu bakalářské práce, panu Ing. Jiřímu Křížovi, Ph.D. za odborné rady při zpracovávání bakalářské práce.

OBSAH:

1. ÚVOD

Ve své bakalářské práci se budu zabývat návrhem databáze pro řízení uživatelských účtů ve společnosti IBM Brno. Zde pracuji v týmu unixových administrátorů, kteří denně řeší různé problémy s uživatelskými účty. Bohužel velká část naší práce končí na administrativě, neboť jsme často nuceni řešit problémy, které již řešeny byly, avšak nikde nejsou evidovány.

Implementace databázových systémů do firem se stává naprostou samozřejmostí, pomáhají například udržovat přehled ve skladech, zjednodušují účetní procesy, zlepšují komunikaci ve firmě atd. S vhodně postaveným informačním systémem dostává firma náskok před konkurencí, ať už zrychlením vyřizování požadavků od klientů, nebo vhodným řízením skladových zásob či zvyšováním výrobních kapacit. K těmto informačním systémům je však zapotřebí mít kvalitní databáze.

Firma IBM ve většině případů používá pro řešení interních problémů vlastní řešení, která neustále vyvíjí, avšak na globální úrovni je většinou problém s jejich implementací. V podstatě je možné říct, že v konečném důsledku IBM nepoužívá žádná globální „end to end“ řešení, ale nechává otevřený prostor jednotlivým pobočkám pro vlastní realizaci. To je například důvodem, proč brněnská pobočka IBM nemá jednotný centralizovaný informační systém, jenž by obstaral veškeré úkoly, které se zde řeší. Například IS pro docházku je zakoupený od jedné společnosti, ale IS pro přidělování práce je převzat z francouzské pobočky. V našem případě vyčleníme aktivitu, která se týká uživatelských účtů do vlastního systému.

2. VYMEZENÍ PROBLÉMU A CÍLE PRÁCE

2.1 VYMEZENÍ PROBLÉMU

Na pozici administrátora unixových systémů řešíme nejrůznější úkoly pro nejrůznější zákazníky z celého světa. V našem týmu se zabývám řešením unixových účtů pro konkrétního zákazníka, jehož jméno z interních důvodů nemohu uvést. Do budoucna by se měla tato aktivita centralizovat pro všechny zákazníky dané země, ovšem to je zatím v nedohlednu. Moje práce se tedy bude týkat velmi úzkého odvětví, neboť v současné době nejsou procesy pro řízení uživatelských účtů (user account management) standardizovány mezi všemi zeměmi a všemi zákazníky.

2.2 CÍLE PRÁCE

Cílem mé práce bude vytvořit databázi, která zefektivní řízení uživatelských účtů, neboť v současné době se jedná o činnost velmi neefektivní. Na této databázi bude fungovat informační systém, jenž umožní trackovat a filtrovat všechny požadavky, týkající se uživatelských účtů. Umožní uživatelům dohledávat jejich žádosti, administrátorům informace o vytvořených účtech a manažerům trackování požadavků o účty. Tento databázový systém přinese i velký přínos do bezpečnosti, neboť bude přesně možné stanovit, kdo bude zodpovědný za vytváření účtů, a kdo za schvalování žádostí. Informační systém bude vytvořen jako webová aplikace, která poběží na serveru s databází. Jako databázový systém byl po konzultaci s administrátory a zákazníkem zvolen MS SQL 2008.

3. ANALÝZA SOUČASNÉHO STAVU

3.1 INFORMACE O SPOLEČNOSTI

3.1.1 Historie a vývoj firmy

Společnost IBM je na trhu IT již od devatenáctého století, konkrétně roku 1888, kdy se zabývala výrobou automatických pokladen, prvních kalkulaček, hodin pro kontrolu pracovní doby a podobně. Během své historie se věnovala vývoji sálových počítačů a od osmdesátých let osobních počítačů. Od devadesátých let se věnuje výhradně službám a navrhování komplexních řešení pro zákazníky jakéhokoliv typu, jako například banky, nemocnice, cestovní kanceláře, obchodní domy a jiné. IBM mělo první pobočku v české republice od roku 1932, to byla v celkovém pořadí šestá evropská země, ve které IBM obchodovala. Tato pobočka však byla po druhé světové válce v roce 1948 znárodněna. Znovu se objevuje IBM v roce 1991 jako československá pobočka. Dnes je sídlem IBM město Armonk v Americkém New Yorku, u nás má IBM pobočky v Praze a v Brně. Generálním ředitelem je v současné době Samuel J. Palmisano, pro českou republiku byl v lednu 2010 jmenován generálním ředitelem Ing. Vladek Šlezingr. Společnost IBM stála u zrodu prvních osobních počítačů, které uvedla na trh v roce 1981. Výroby osobních počítačů se vzdala až v roce 2004, kdy prodala poslední část výrobní divize čínské firmě Lenovo. Od té doby vyrábí už pouze servery a nabízí veškeré služby s nimi spojené.

Společnost IBM je v dnešní době špičkou na trhu v navrhování a udržování komplexních IT řešení pro firmy podnikajících v jakémkoliv odvětví. To jí přináší vyšší zisky než osobní počítače, protože outsorcované služby mají vyšší přidanou hodnotu. Pouze příjmy z minulého roku mělo IBM ve výši kolem 95 miliard amerických dolarů. V této práci se budu zabývat pouze malým zlomkem pracovní činnosti společnosti IBM, a to pobočkou IDC Brno, kde se zabýváme strategickým outsorcingem - sledováním a údržbou systémů, vzdálenou správou serverů, podporou uživatelů, a řešením problémů, které mohou ohrozit produkční činnost zákazníka.

3.1.2 Swot analýza

SWOT analýza je soubor nástrojů, používaných k ohodnocení jednotlivých ukazatelů ve společnosti, které jsou vhodné zejména pro strategické plánování firmy nebo i jednotlivých částí firmy. Zde budu vztahovat analýzu pouze na část firmy, která se zabývá řízením a podporou uživatelských účtů.

Silné stránky: Zkušení administrátoři a lidé, vykonávající samotnou podporu.
Slabé stránky: Nevhodný informační systém, který nereflektuje potřeby bezpečnosti.
Příležitosti: Zvýšení kvality poskytovaných služeb, možnost rozšíření navrhovaného systému k dalším zákazníkům.
Hrozby: Možnost vzniku bezpečnostního incidentu, a to kvůli nemožnosti kontroly nad uživatelskými účty. Vysoké sankce, možnost výpovědi smlouvy ze strany zákazníka.

3.1.3 Správa uživatelských účtů

Správné řízení uživatelských účtů je velice diskutabilní otázka. Živnostník, který má na firemním počítači jeden uživatelský účet, se touto problematikou příliš zabývat nemusí, ovšem velké firmy, kde jsou desítky, případně stovky zaměstnanců, se touto problematikou zabývat musí. Pokud se bavíme o firmách, pracujících v IT (nebo s IT), tak obecně platí, že čím více má firma zaměstnanců, tím více se musí starat o jejich uživatelské účty. Ve velkých společnostech je těžké, ne- li dokonce nemožné, uhlídat, kolik je kde vytvořených uživatelských účtů a zda nedochází k potížím s bezpečností z důvodu neoprávněných přístupů. S tím souvisí samotné žádosti o vytvoření účtu nebo žádosti o přidělení různých přístupových práv. Tyto žádosti většinou vznikají odděleně (od manažera, od HR, od samotného zaměstnance, či jen požadavkem přes interní poštu), často bez vědomí nebo bez kontroly bezpečnosti manažerem či jiným zaměstnancem, hlídajícím oprávněné žádosti.

Kvůli těmto faktům jsou firmy, které mají podobné problémy, v podstatě nuceny zavádět nějaká centrální řešení, jež pokryjí všechny potřeby firmy, uživatelů, sítě, bezpečnosti atd.

Tato celková řešení se většinou neimplementují do začínající firmy, většinou se začínají využívat až v momentě, kdy firma (nebo objem dat, účtů) vyroste do takové výše, kdy je nutné začít dělat přehled. Jejich snahou není měnit organizační strukturu firmy či již udělené uživatelské přístupy. Implementace by měla probíhat dle potřeb společnosti s postupně od nejcitlivějších oblastí (aplikační účty, důležité účty operačního systému, data uživatelů) a poté začít implementovat další oblasti podle předem stanovených priorit. Takové řešení by mělo být navrženo tak, aby bylo flexibilní a mohlo se přizpůsobovat neustále se měnícím požadavkům.

3.2 POPIS SOUČASNÉHO STAVU

3.2.1 Řešené problémy

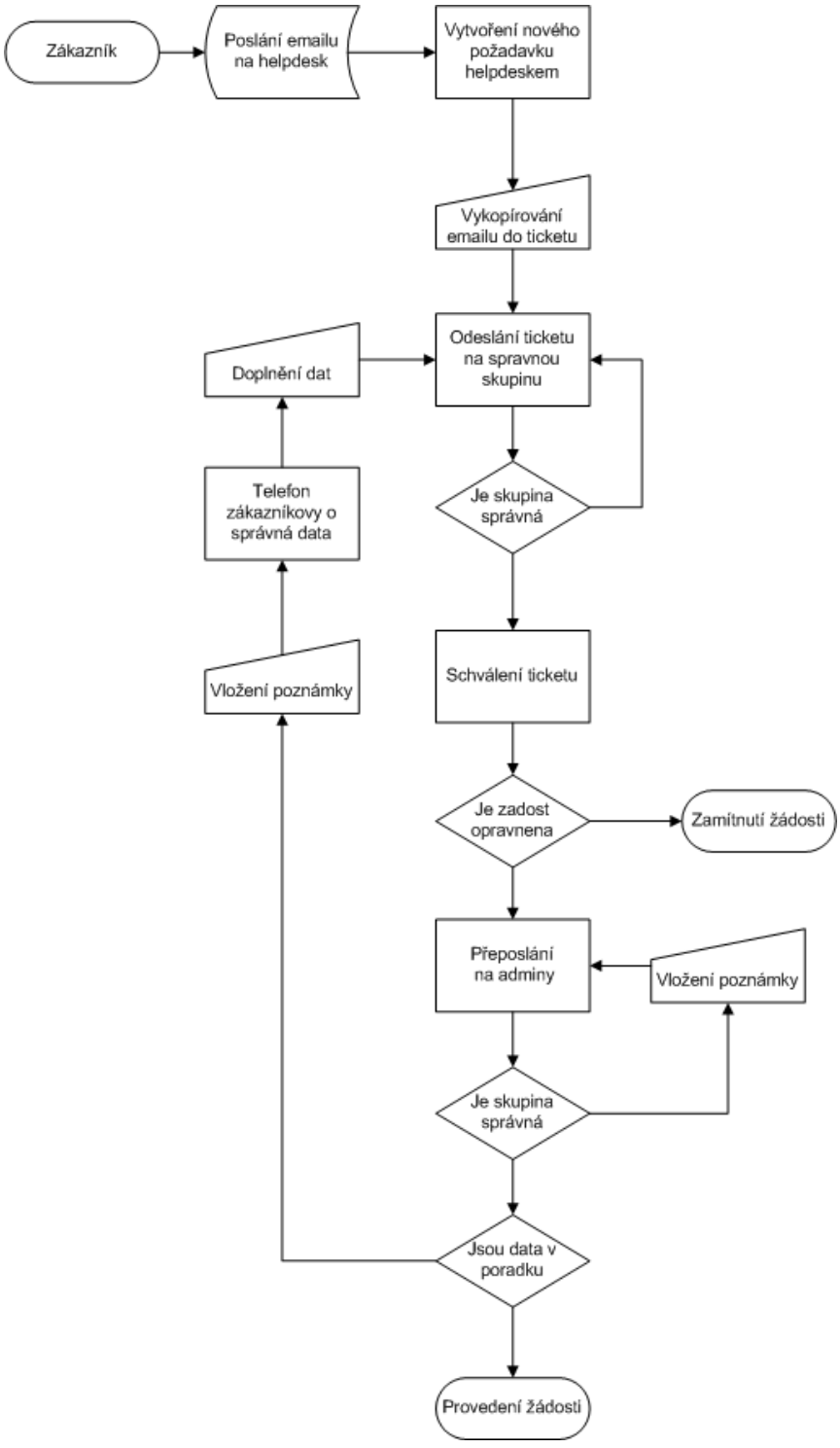
Správa uživatelských účtů (user account management) spočívá ve vytváření uživatelských účtů na asi stu zákaznických serverech, přidělováním skupin, kde skupiny řeší, jaká práva má uživatel ke konkrétním aplikacím či adresářům. Dále přidělováním tzv. “super RIGTS”, které opravňují uživatele používat některé aplikace či speciální účty. Resetování hesel, které provádíme v případech expirace uživatelského hesla nebo v případě, že uživatel heslo zapomene či ztratí. Dále se věnujeme modifikacemi uživatelských účtů a odstraňováním účtů ze serverů. Všechny tyto činnosti provádíme ze systému windows pomocí unixové konzole na serverech s různými unix systémy jako aix, solaris, linux atd. Požadavky na tyto činnosti jsou zasílány formou ticketu na konkrétní zodpovědnou skupinu, která má za úkol daný problém v daném čase vyřešit. Tyto podmínky jsou dány smlouvou mezi zákazníkem a společností, udržující vzdálenou správu.

3.2.2 Popis stávajícího informačního systému

Manage Now je webový produkt, který je součástí Enterprise System Management. Toto řešení je postaveno tak, aby se v nich daly řešit jakékoliv vztahy mezi uživateli, změny v informačních systémech, podnikové aktivity atd. Skládá se ze třech částí: Problem management, Change management, Asset management. V podstatě se jedná o nástroj na delegování činností, respektive přeposílání informací s určitou prioritou mezi jednotlivými týmy.

Nadále budu popisovat pouze část aplikace zvanou "Problem management"

Problem mangement je aplikace pro řešení problémů ve firmě. Problém je definován jako událost, která může znamenat ztrátu dostupnosti dat či výkonu serverů nebo snížení kvality poskytovaných služeb. To zahrnuje chyby, související s hardwarem, softwarem nebo aplikacemi různých systémů, sítí, atd. Problem management též zjednodušuje proces registrace hovorů zákazníka na helpdesk tím, že pracovníkům podpory dává přístup k rozsáhlé a neustále se rozvíjející databázi problémů a jejich řešení, proto je možné dle čísla ticketu (problému) dohledat jakoukoli situaci z minulosti a navrhnout okamžité řešení, případně kontaktovat ty správné osoby.



Obrázek 1: *Vývojový diagram současného procesu.*

3.2.3 Analýza databáze současného řešení

Manage Now je vlastně fronta, do níž chodí požadavky na provádění činnosti. Je vázán na osobu, která se do něj při spouštění přihlašuje. Fronta, kterou může uživatel zobrazit, je vázaná na skupinu lidí s autorizovaným přístupem do této skupiny. Bohužel je vše řešeno pomocí textových polí, proto není možné vytvářet přehlednou databázi

uživatelů a serverů pro konkrétního zákazníka. Častým problémem je, že zákazník neví, na jakých serverech chce mít účet a s jakými právy, neboť si je nemůže vylistovat přímo v žádosti a musí mít tyto informace z jiných zdrojů, které jsou tvořeny různými excelovými soubory a podobně. Výhoda spočívá v přeposílání jakýchkoliv úkolů mezi jakýmikoliv skupinami a možností vkládání příloh například se seznamem serverů. R2 Manage Now je vázaná na skupiny, takže není podstatné, kdo žádost posílá, ale zda ji posílá na správnou skupinu. Dochází často k chybám se zasíláním ticketů, neboť ne vždy je problém zaslán na správnou skupinu. Tyto chyby vznikají na mnohých místech. Nejčastěji jde však o neinformovanost uživatelů v tom, na koho se mají obrátit s netypickým problémem.

3.2.4 Výhody a nevýhody stávajícího řešení

Výhody:

- Účinné a komplexní řešení
- Rychlé a přehledné
- Vhodné pro předávání jednoduchých úkolů či informací
- Sběr všech používaných dat
- Praktické
- Hlídání kvality pomocí SLA
- Vhodné pro různorodé problémy

Nevýhody:

- Nízká možnost filtrace sesbíraných dat
- Naprosto nevhodné pro řízení uživatelských účtů
- Žádosti chodí v podobě textových polí

3.2.5 Vyhodnocení analýzy

R2 Manage Now je sice výkonné řešení, jehož hlavní výhodou je univerzálnost, rychlost a jednoduchost, avšak tyto výhody se v zápětí stávají nevýhodami při potřebě sledovat předchozí žádosti. V případě, že si uživatel zažádá o vytvoření účtů, je nemožné zjistit z předchozích žádostí, zda už si někdy o účet žádal či nikoliv. Tím vznikají problémy, že si uživatel žádá několikrát, přitom však potřebuje password reset. Nebo naopak chce přidat práva k různým aplikacím, které už má, nebo si žádá o práva tam, kde nemá účet. Tím se zpomaluje celková činnost produkce. Oddělení této aktivity z informačního systému Manage Now a ukládání informací o uživatelích, jejich účtech a serverech do vlastní databáze by problém vyřešilo.

4. TEORETICKÉ ŘEŠENÍ

4.1 INFORMAČNÍ A DATABÁZOVÉ SYSTÉMY

4.1.1 Informační systém

Informační systémy plní úlohu sběru a zpracovávání informací dle procesů v podniku. Informační systémy pracují s daty, které oraganizují a třídí a tím vznikají informace. Daty myslíme jakékoliv zaznamenané poznámky, soubory, obrázky, které bez správného zpracování neposkytují žádné informace. Informacemi myslíme jakákoliv sdělení, kterými odstraňujeme nejistotu či nevědomost a které jsou pro uživatele hodnotné a přínosné. Historicky byly informační systémy v podobě kartoték, účetních knih a různých seznamů. V dnešní době se za informační systém obecně považuje podnikový software, jenž automatizuje podnikové procesy nebo jinak pomáhá běhu a řízení podniku. (5)

Informační systém může být programován jako samotná aplikace nebo jako v poslední době stále oblíbenější používání webových aplikací, vytvořených v jazyce java, php či asp .NET.

4.1.2 Databázový systém

Databázový systém je tvořen dvěma základními složkami: systémem řízení báze dat (SŘBD) a databází. Data, která soustředíme do nějaké databáze, vytváří nějaké databázové schéma. Toto schéma obsahuje metainformace čili informace, jež popisují data. To znamená, že nám říkají, jak mají data spravně vypadat. Schéma databáze dále popisuje objekty a vztahy mezi nimi. Tyto objekty se liší podle toho, na jakém SŘBD je systém založen. Datové modelování je prostředek k vytvoření schématu databáze, je to jakási kolekce pojmů, na niž je vytvořen jazyk pro definici dat. (5)

Na trhu existuje mnoho různých systémů pro řízení báze dat. Jako příklad bych rád uvedl několik nejznámějších, jako MySQL, ORACLE, DB2, MS SQL. V následujícím odstavci uvedu několik informací o MS SQL 2008

MS SQL 2008 je komplexní nástroj, umožňující sdílet data uživatelům kdykoliv a kdekoliv. Umožňuje ukládat všechny typy dat - data strukturovaná, částečně strukturovaná, nestrukturovaná.SQL SERVER umožňuje provádět vyhledávání dat, synchronizaci dat, analýzu, podávat různé dotazy atd. Tento software nabízí komplexní nástroje pro business inteligence, data je pak možno prohlížet pomocí běžných nástrojů Microsoft Office. MS SQL je nástroj, ve kterém je možné vytvářet databáze,

multidimenzionální databáze, provádět dotazy nad databázemi a stavět nad nimi informační systémy.(6)

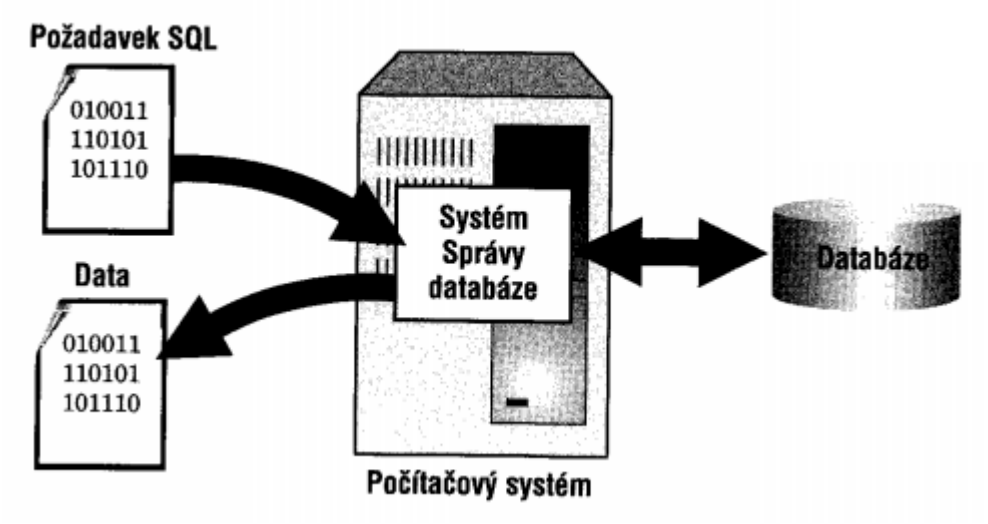
4.2 SQL JAZYK

Jeho historie spadá do 60. až 80. Let, kdy se ve firmě IBM objevily první snahy o databázové zpracování dat. Tento model byl založen na hierarchické struktuře, jež vyházela ze struktury firemní organizace. První standart byl stanoven v roce 1986 jako SQL86. Původní snahou bylo vyvinout jazyk, jenž bude jednoduchý a v podstatě podobný angličtině. Je jasné, že tento původní záměr byl nereálný, avšak stejně je to jazyk relativně jednoduchý. Po opravení různých problémů a nedostatků byl stanoven standart SQL92, na který navazuje SQL99, reagující na potřeby využívání objektových prvků v databázích. SQL99 je platný dodnes. SQL znamená strukturovaný dotazovací jazyk (structured querry language). Jazyk je určen pro správu tabulek, databází a pro práci s daty v nich obsaženými. Používá se k ovládání prakticky všech relačních databází.

```
SELECT·customers.*·FROM·customers·
LEFT·JOIN·orders·ON·
(customers.customer_id·=·
orders.customer_id·AND·
year(orders.order_date)·=·2004)·
WHERE·orders.order_id·IS·NULL¶
```

Obrázek 2: *Příklad SQL dotazu*

Jazyk SQL přistupuje k datům pomocí databázového systému neboli DŘSB. V jazyce SQL napíšeme požadavek na data, jež nás zajímají, databázový systém jej načte, zpracuje, a vrátí odpovídající informace. Proces, kterým požadujeme data z databáze, se nazývá dotaz. Odtutd plyne název SQL- strukturovaný dotazovací jazyk. (4)



Obrázek 3:*Třívrstvý databázový systém. Převazato ze (4).*

4.3 DATOVÉ MODELY

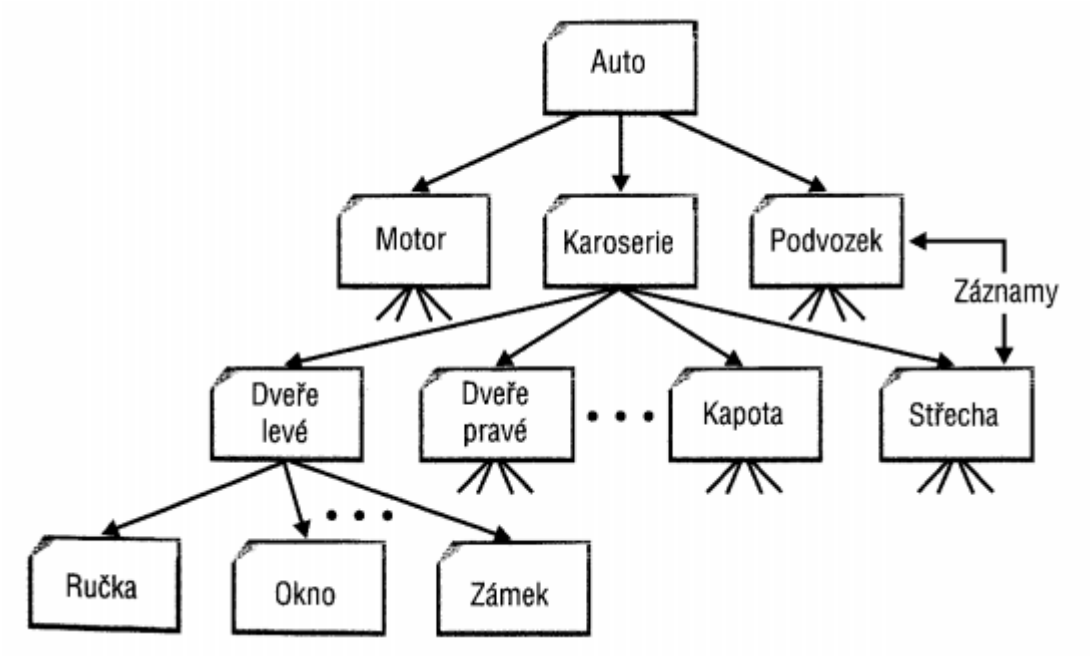
Než začneme navrhovat nějakou datovou strukturu, musíme zvolit vhodný model pro ukládání různých datových objektů. V sedmdesátých až osmdesátých letech se objevilo několik datových modelů, které měli své výhody i nevýhody. Hlavním cílem je vytvořit takový obraz reality, aby data, vložená do informačního systému, této realitě o nejvíce odpovídala. SQL je databázový jazyk, jenž je postavený na relačním datovém modelu. Abychom si vytvořili představu o tom, jak relační model vznikl, popíši některé dříve používané datové modely.(4)

4.3.1 Lineární datový model

Je to jediný datový model, který nereflektuje žádné vazby, ale je možné ho implementovat na libovolné médium. Vhodným příkladem je kartotéka zaměstnanců, kde jednotlivé karty s údaji o zaměstnancích jsou uloženy v krabici. Každá karta tvoří “řádek” v databázovém souboru, a každá krabice tvoří databázový soubor.(2)

4.3.2 Hiearchické databáze

Jendou z nejvýznamějších aplikací pro první databázové systémy bylo plánování výroby. Pro zoodpovězení různých otázek segmentu plánování byla vytvořená hierarchická struktura seznamu dílů, součástek, komponent, které se říkalo kusovník. K ukládání kusovníku byl vyvinut hiearchický datový model.

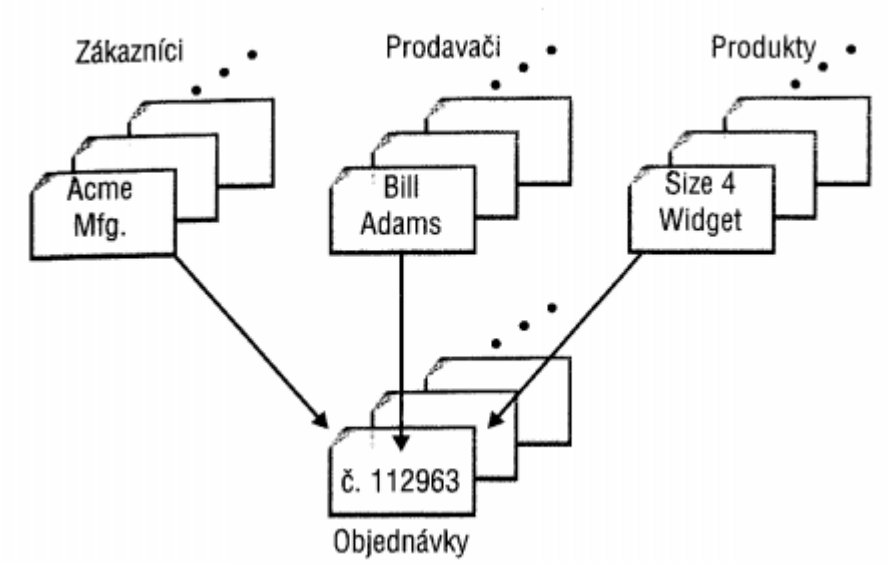


Obrázek 4:*Hierachický datový model. Převzato z (4)*

Hierarchický datový model je tvořen segmentem (rodič), ze kterého vedou vazby na jemu podřízené segmenty (potomky). Vazby na jednotlivé segmenty jsou vedeny tzv. poinetry, jež tvoří databázový systém, na kterém je model implementován. Výhodou byla rychlost při vyhledávání požadovaných údajů a přehlednost, nevýhodou dlouhá doba, nutná na reorganizace databázových souborů. Tento systém je dodnes používán na mainframech od IBM, neboť je velmi vhodný pro zpracovávání velkého množství transakcí.(4)

4.3.3 Síťové databáze

Síťový a hierarchický model jsou si celkem podobné. Hierarchický model byl nevýhodný ve chvíli, kdy měla data složitější strukturu. Proto se začaly vytvářet pointery i mimo vztahy rodič - potomek. Tyto vazby se u síťového modelu označují jako vztahy. Pro navigaci v těchto databázích se užívalo nejen směru, jak je to v modelu hierarchickém, ale také relaci, na kterou je třeba se přesunout. Databáze byly téměř tak výkonné jak hierarchické, ale měly větší podporu pro reprezentaci dat díky většímu množství vazeb. Mezi hlavní nevýhody opět patří nepoddajnost při reorganizaci dat a nutnost dopředu specifikovat sady relací a strukturu databáze. (4)

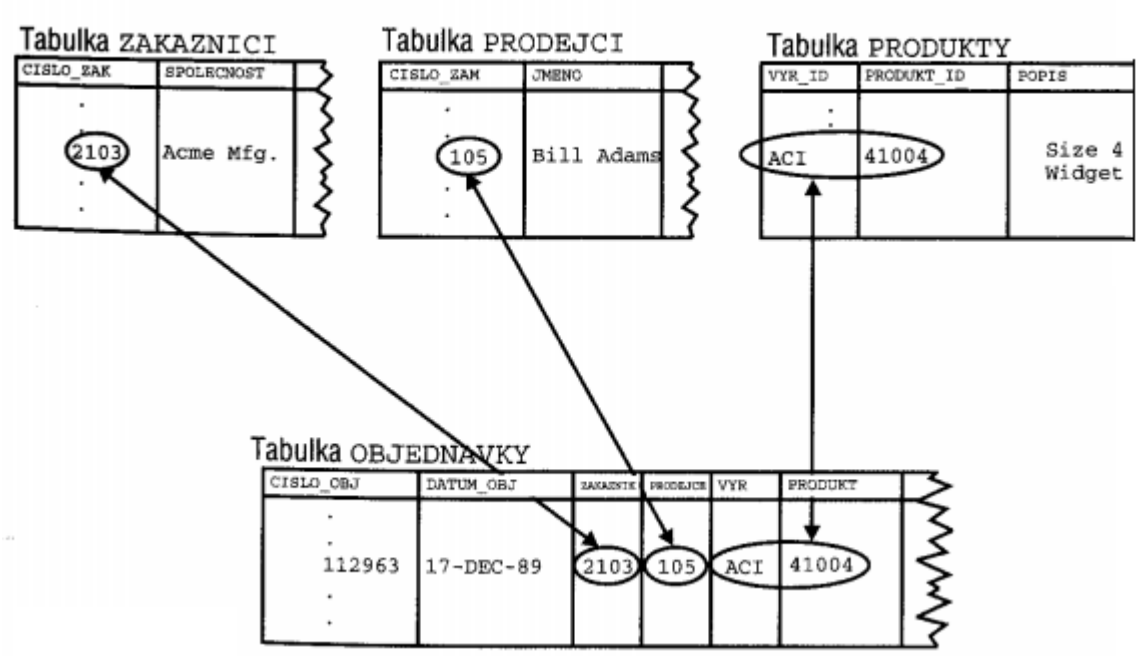


Obrázek 5:*Síťový datový model. Převzato z (4)*

4.3.4 Relační datový model

V roce 1970 vytvořil doktor Codd relační datový model. Ten se rozšířil až polovině osmdesátých let jako pokus o zjednodušení struktury databáze. Eliminoval všechny vztahy potomek - rodič a místo toho začal data ukládat do prostých tabulek . V průběhu vývoje vznikaly různé databázové systémy, které se označovaly jako relační, ale selhávaly při implementaci určitých částí Coddova modelu. Tím se stalo nejednoznačné, co je a co není relační databáze, proto Dr. Codd v roce 1985 napsal 12 Coddových pravidel, kterými se musí databáze řídit, pokud mají být skutečně relační. Z těchto pravidel se později vyvinuly normální formy. Ty jsou popsány níže.

Základním organizačním prvkem relační databáze je tabulka. Tabulka má v rámci databáze unikátní název, který identifikuje její obsah. Každý sloupec v tabulce má svůj název, avšak není vyloučeno, aby byly dva stejné sloupce v rámci dvou různých tabulek. Jelikož nejsou řádky tabulky uspořádány, není možné identifikovat například sedmý řádek, první řádek atp. V relační databázi má každá tabulka sloupec nebo kombinaci několika sloupců, jejichž hodnoty jednoznačně identifikují řádek v tabulce. Tento sloupec se nazývá primární klíč. Pro propojení více tabulek využíváme takzvaných cizích klíčů. Cizí klíč je tedy sloupec v jedné tabulce, jehož hodnota odpovídá primárnímu klíči v jiné tabulce. Cizích klíčů může být v jedné tabulce několik, avšak primární může být pouze jeden.(4)



Obrázek 6: *Relační datový model. Převzato z (4)*

4.4 WEBOVÁ APLIKACE

4.4.1 Jazyk HTML

O jazyce HTML se zmíním jen stručně, neboť je třeba znát pro vývoj webových aplikací pouze jeho základy. První návrhy HTML (HyperText Markup Language) jsou z roku 1989. První verze však byla dokončena až v roce 1993. O vývoj jazyka HTML se stará W3 consorcium v rámci nějaké standartizace, neboť různí uživatelé začali vytvářet vlastní popisovací tagy a podobně. Poslední verze HTML je 4.01. Ta byla vydaná v roce 1999. Nyní se pracuje (již několik let) na vývoji HTML verze pět.

Jazyk HTML má jasně definovanou syntaxi a internetovému prohlížeči vlasntě říká, jak má zpracovat určitou značku (tag) a jak tato značka bude interpretována na monitor. Tyto značky lze z hlediska významu rozdělit na tři skupiny :

Strukturální značky: vytvářejí strukturu dokumentu, tzn. Odstavce<p>, nadpisy<h1>,<h2>...

Popisné značky: popisují povahu obsahu elementu, jako například nadpis, adresa, data. Tyto snahy o co nejrozumější popsání vedly k rozmachu a vývoji jazyka XML.

Stylistické značky: určují vzhled jednotlivých elementů, jako například tučné písmo, barvu písma, zarovnání atd. Dnes se od jejich používání ustupuje k používání kaskádových stylů, které mají nezbytnou výhodu v možnosti uložit několik typů zobrazení. (7)

4.4.2 Kaskádové styly – CSS

CSS (cascading style sheets) vzniklo v roce 1997 jako součást HTML 4. CSS umožňuje definovat vlastnosti určitých elementů, jako je barva pozadí, přechody, velikosti písem, nadpisy, rozvržení stránky a další vlastnosti, které nejsou v HTML možné, případně je zbytečné je pořád vypisovat do každé stránky. V CSS se jednoduše vytvoří předloha pro vzhled stránky, na niž se pak ve všech dalších stránkách odkazuje a načítá si vzhled. To přináší výhodu především při aktualizaci stránek - aktualizuje se pouze jedna. Další velmi zajímavou výhodou je zvýšení rychlosti, neboť při načtení se styl stáhne pouze jednou, poté zůstává v cash paměti.

CSS se připojí k HTML stránce následujícím příkazem, jenž je zapsán do HTML kódu:

```
<link rel="stylesheet" type="text/css"href="styl.css">
```

Obrázek 7: *Připojení CSS k HTML stránce*

4.4.3 PHP – personal home page (tools)

PHP, jak je známe dnes, vzniklo až v roce 2005 a oficíalně se prezentuje jako PHP verze 5. Avšak první zmínky o jazyce php se objevily již v roce 1995, což byla spíše sada skriptů v jazyce perl pro zpracování záznamů a přístupů k webu. Protože bylo zapotřebí větší funkčnosti, byla vytvořena mohem rozsáhlejší implementace v jazyku C, která uměla komunikovat s databázemi a vytvářet jednoduché dynamické aplikace pro web. Postupně se shledávalo, že je PHP poddimenzovaný pro vývoj aplikací pro e-business. PHP dnes je skriptovací jazkyk, který se používá jak pro vývoj dynamických webů, tak i k vývoji konzolových a desktopových aplikací.

Skripty PHP jsou zpracovávány na straně serveru, proto se k uživateli dostane až výsledek zpracování (při prohlížení zdrojového kódu stránky nenalezneme, co napsal programátor, ale již serverem zpracovaný výstup). PHP má obrovskou výhodu, že je nezávislý na platformě, proto se hojně používá na všech serverových systémech. Další výhodou je skutečnost, že je tento jazkyk zdarma, tím se stal v podstatě standartem, který podporuje většina serverů.

PHP kód se vkládá přímo do HTML kódu stránky a je oddělěn těmito znaky:

"<? *php kód* ?>"

cokoliv je mimo tento kód se považuje za HTML kód.

4.5 VYTVÁŘENÍ DATABÁZE

4.5.1 Relace

Tabulka v databázi. Její sloupce jsou atributy, její řádky záznamy. Jako příklad atributu může být barva (například automobilu) a jeho záznam je například zelená.

4.5.2 Primární klíč

Jednoznačný nenulový identifikátor záznamu v tabulce. Primární klíč slouží k adresování n-tic relace. Příklad primárního klíče v tabulce automobilů může být například SPZ.

4.5.3 Cizí klíč

Sloupec jedné tabulky ukazující na primární klíč jiné tabulky. Společně s primárním klíčem tedy vytváří vazbu mezi tabulkami.

4.5.4 Normalizace tabulek

Normalizace dat je činnost, která optimalizuje návrhy datových struktur tak, aby splňovaly zvolené normalizační formy, repsektive úrovně. Tyto formy jsou navrženy tak, aby splňovaly požadavky na efektivní ukládání dat a minimalizovali redundanci při zachování integrity a konsistence dat. Pokud některý datový model porušuje některou z normalizačních forem, není navržen optimálně. Pokud chceme databázi normalizovat na vyšší úroveň, musí být splněná normalizace i na všech předchozích úrovních.(2)

Normální formy lze rozdělit do dvou skupin: Jednu skupinu tvoří první tři normální formy, které jsou součástí Coddovy relační teorie. Druhou skupinu tvoří Boyce Coddova, čtvrtá a páta normálni forma. Pro běžné účely stačí první tři normální formy.

1.normální forma -multizávislost:

Relace je v první normální formě, pokud jsou všechny její atributy definovány nad skalárními obory hodnot (doménami). Tabulka je tedy v první normální formě, pokud lze do jednoho pole dosadit pouze jeden datový typ. To znamená, že hodnoty jsou dále nedělitelné. (2)

2.normální forma -funkční závislost:

Relace je ve druhé normální formě, pokud je v první normální formě a navíc všechny její atributy jsou závislé na celém kandidátním (primárním) klíči.(2)

3.normální forma -tranzitivní závislost:

Relace je ve třetí normální formě, pokud je ve druhé normální formě a navíc všechny její neklíčové atributy jsou vzájemně nezávislé. (2)

Boyce Coddova normální forma:

Též se nazývá variací třetí normální formy. Relace je ve Boyce-Coddově normální formě, pokud mezi kandidátními klíči není žádná funkční závislost, a to za následujících třech podmínek:

relace musí mít dva nebo více kandidátních klíčů
nejméně dva z kandidátních klíčů musí být složené
kandidátní klíče se v některých atributech musí překrývat (2)

4. normální forma:

Relace je ve čtvrté normální formě, pokud je v Boyce-Coddově normální formě a navíc všechny vícehodnotové závislosti jsou zároveň funkčními závislostmi z kandidátních klíčů (v jedné relaci nesmí docházet ke spojení nezávislé opakované skupiny). (2)

5.normální forma:

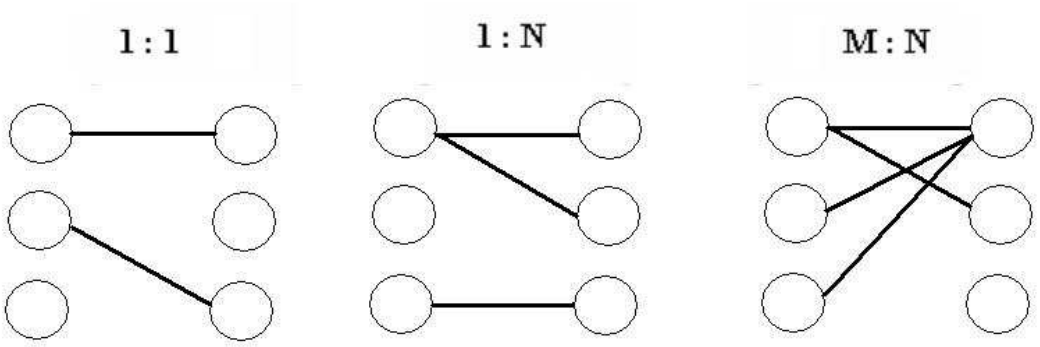
Řeší případy společné závislosti, která vyjadřuje cyklické omezení: pokud je relace 1 spojena s relací 2, relace 2 je spojena s relací 3 a relace 3 je zpětně spojená s relací 1, pak všechny tři entity musí být součástí stejného vektoru hodnot. (2)

Tabulky by měly být vytvářeny alespoň ve třetí normální formě.

4.5.5 Vazby mezi entitami

Vazby mezi entitami představují logické vztahy mezi entitami. V datových modelech jsou povoleny pouze vazby mezi dvěma entitami – binární vazby. Na vazbu pohlížíme jako na dvě vazby v opačných směrech. V tomto případě mluvíme o takzvaných, rolích které představují vazby mezi entitami ve směru od jedné k druhé. Ke každé z rolí přiřadíme takzvanou kardinalitu. Ta představuje omezení v počtu instancí druhé entity, jež má vztah s jakoukoliv instancí první entity. Minimální kardinalita může být 0 nebo 1, maximální kardinalita může být 1 nebo N. (3)

1:1 - jednomu záznamu jedné tabulky odpovídá právě jedna položka druhé tabulky
1:N - jednomu záznamu jedné tabulky může odpovídat více položek druhé tabulky
M:N - více záznamům jedné tabulky může odpovídat více položek druhé tabulky



Obrázek 8: *Kardinalita databázových vztahů. Převzato z (2)*

Při návrhu databáze musíme uvažovat, k jakému účelu bude používána a jaké atributy se v ní budou vyskytovat, jaké elementy. Nejlepší je postupovat od grafického návrhu na papír až k samotnému softwarovému řešení.

Jako první krok je doporučováno stanovit, jaké tabulky se v databázi budou vyskytovat. Tabulky je nutné vymýšlet tím způsobem, aby odpovídaly normálním formám a aby neobsahovaly příliš mnoho redundantních informací. V dalším kroku je nutné stanovit, jaké budou tabulky obsahovat atributy. U každého atributu je třeba vhodně zvolit datový typ, aby byla výsledná databáze optimalizovaná, to znamená, aby byla rychlá a nezabírala zbytečně mnoho místa. V třetím kroku je nutné určit primární klíč každé tabulky a začít přemýšlet, jak vyřešit vztahy mezi relacemi pomocí cizích klíčů. Relační databáze v podstatě podporují pouze vztahy 1:N, proto pokud se nám stane, že musíme vyřešit vztah M:N, je nutné zpracovat pomocnou tabulku na dekompozici na dva vztahy 1:N. Pokud máme hotový návrh databáze, můžeme se pustit do samotného psaní kódu.

4.5.6 Datové typy

Textové:

CHARACTER (n) řetězec libovolných znaků o délce 1-255

CHARACTER VARYING, pole znaků proměnné délky; VARCHAR (n)

Číselné:

NUMERIC (x,y) reálné číslo

DECIMAL; desetinné číslo s pevnou řádovou čárkou ; DEC (x,y)

INTEGER celé číslo včetně znaménka o délce 8 bajtů ; INT

SMALLINT celé číslo včetně znaménka o délce 4 bajty

Datové a časové údaje:

DATE kalendářní datum

TIME denní čas

TIMESTAMP časové razítko

INTERVAL časový interval

Ostatní datové typy:

BOOLEAN logické hodnoty true/false/unknown

BLOB rozsáhlý binární datový objekt – obrázek, zvuk, soubor

4.5.7 SQL příkazy

Sql příkazy se dělí do čtyř následujících skupin:

Data definition language : jedná se o jazyk pro vytváření databázových schémat a katalogů.

Storage definition language:definuje způsob ukládání dat

View deifinition language: určuje vytváření pohledů

Data manipulation language: obsahuje příkazy pro manipulování s daty. (2)

Nyní uvedu pro příklad několik příkazů včetně jejich syntaxe pro zápis v SQL.

DDL:

Vytvoření databáze:

```
Create DATABASE database_name
```

Vytvoření tabulky:

```
CREATE TABLE table_name
(
column_name1 data_type,
....
)
```

DML:

Vložení hodnot do tabulky:

```
INSERT INTO table_name
VALUES (value1, value2, value3,...)
```

Výběr dat z tabulky:

Jak lze vidět, MS SQL 2008 pracuje s širšími možnostmi nastavení databáze oproti samotnému SQL:

```
USE master
GO
CREATE DATABASE Sales
ON
( NAME = Sales_dat,
   FILENAME = 'c:\database\saledat.mdf',
   SIZE = 10,
   MAXSIZE = 50,
   FILEGROWTH = 5 )
LOG ON
( NAME = 'Sales_log',

   FILENAME = 'c:\database\salelog.ldf',
   SIZE = 5MB,
   MAXSIZE = 25MB,
   FILEGROWTH = 5MB )
GO
```

Obrázek 9: *Vytvoření databáze v MS SQL 2008*

Toto by měly být dostatečné informace pro orientaci v návrhu databáze a můžeme tedy realizovat vlastní vytvoření databáze pomocí SQL.

4.6 IMPLEMENTACE DATABÁZE

Implementace databáze spočívá v naplnění výchozích dat, se kterými bude databáze nutně pracovat. Další částí je vývoj informačního systému nebo aplikace, jež bude využívat vytvořenou databázi. Databázi je nutné umístit na databázový server, kam se budou přihlašovat uživatelé z klientských počítačů. Přistupovat k datům se tedy bude přes informační sysém, který je přímo napojen na databázový řídící systém. Potřebujeme-li z databáze získávat data, napíšeme v jazyce sql dotaz. Datbázový systém jej načte, zpracuje a vrátí požadované informace. (4)

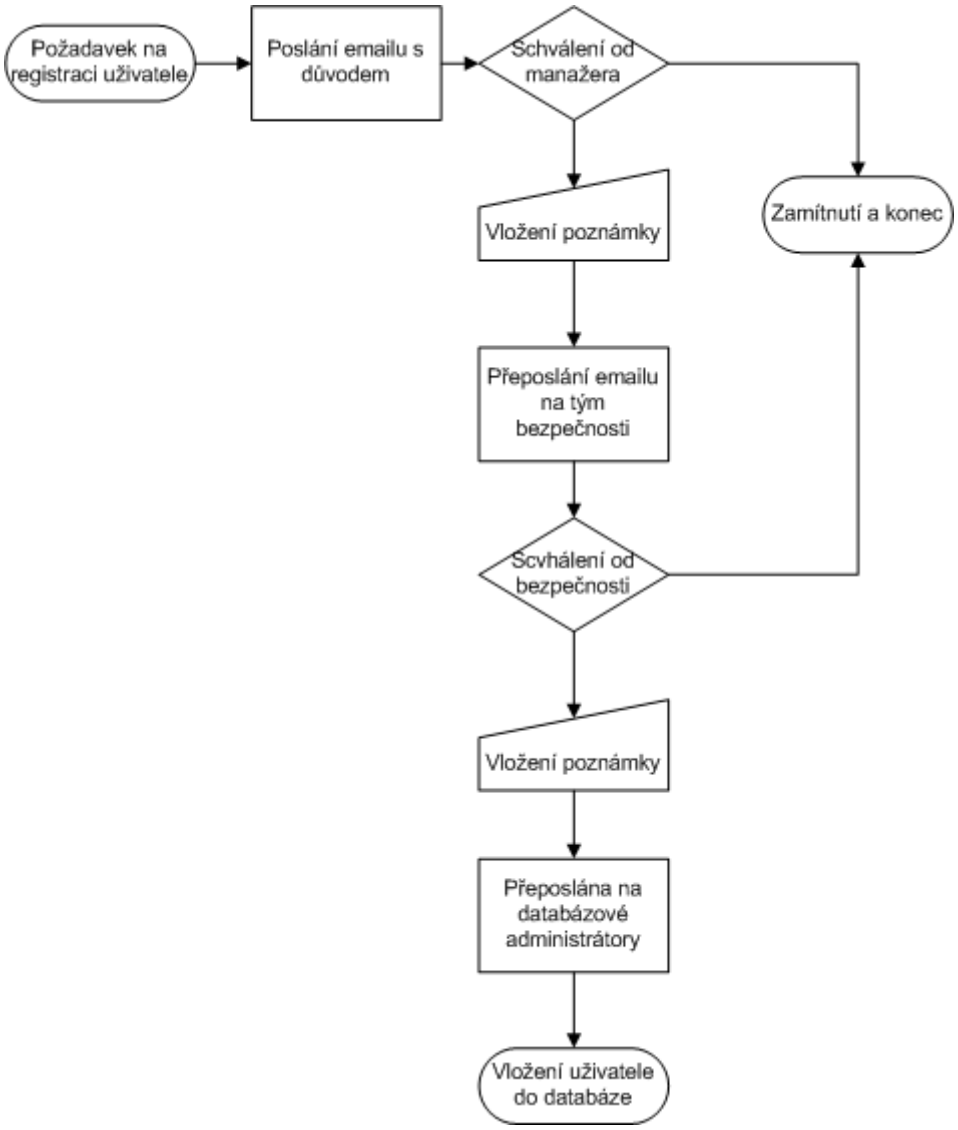
5. NÁVRH ŘEŠENÍ SOUČASNÉ SITUACE

5.1 POHLED NA DATABÁZI Z HLEDISEK BUDOUCÍHO IS

Řešení problému s uživatelskými účty bude spočívat ve zpracovaní nového informačního systému, optimalizovaného pro řízení uživatelských účtů. Informační systém bude fungovat jako jednoduchý objednávkový systém, který bude mít předdefinovaná určitá data, ze kterých uživatel vybere podle svých požadavků to, co potřebuje ke své práci.

Vybírat bude moci servery, na které žádá o přístup a zároveň práva nebo skupiny, případně obojí. V samotné databázi tedy budou obsaženy tři tabulky konečných dat, tedy servery, skupiny, práva, ostatní data budou s časem užívání narůstat na objemu. Dále bude nutné dostat do informačního systému uživatele. IS nebude podporovat automatickou registraci, a to zejména kvůli bezpečnosti. Nový uživatel tedy bude vložen do databáze manuálně administrátorem databáze na základě požadavku od manažera. To především kvůli bezpečnosti. Uživatele rozlišujeme v rámci firmy pouze podle zaměstnaneckého čísla a pobočky, ve které pracuje, čili země. Proto je nutná tabulka zemí. Dále je nutná tabulka samotných ticketů, které tvoří jakousi objednávku od uživatele

Náš systém bude mít tři moduly - jeden manažerský, v němž se bude pouze schvalovat, zda je žádost oprávněná. Druhý modul bude uživatelský - v něm si uživatel požádá o servery, které potřebuje. Třetí modul bude administrátorský. Ten bude zobrazovat frontu ticketů, jež mají být zpracovány. V každém modulu bude možnost vyhledávání ticketů, uživatelů či jakýchkoliv dalších informací, které jsou uloženy v naší databázi. Pro oddělení možnosti nalogování do těchto modulů je třeba vytvořit tabulku funkcí, kde definujeme, jakou roli bude uživatel našeho systému mít.

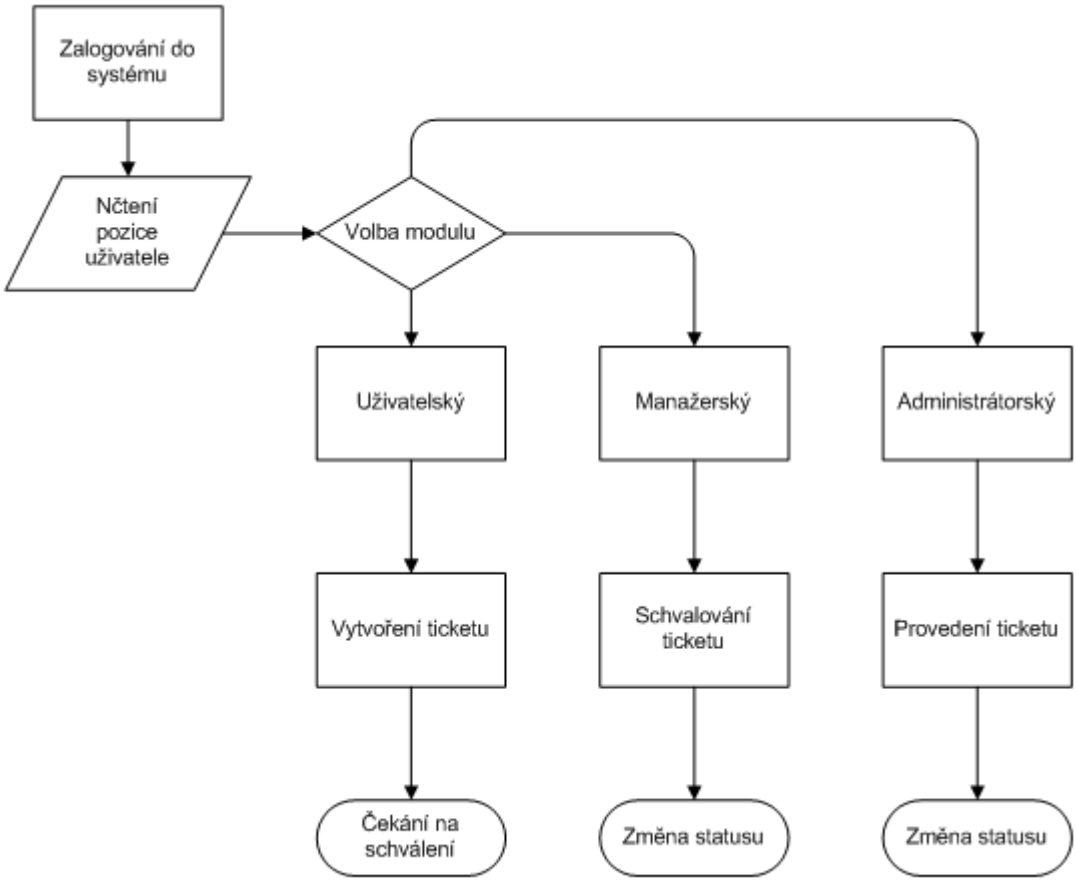


Obrázek 10:*Vytvoření nového uživatele Informačního systému.*

Systém by tedy měl mít tyto funkce:

- Přihlášení do systému
- Vytvoření ticketu
- Výběr serverů, skupin, práv
- Uložení ticketu
- Přeposlání ticketu do dalšího statusu
- Schvalování
- Zavření ticketu
- Vyhledávání

5.2 VOLBA MODULU IS



Obrázek 11:*Automatický výběr uživagtelského modulu*

Informační systém si při logování uživatele stáhne z databáze informace o pozici, na niž se nachází. Dle toho zvolí modul, který bude uživateli zobrazen. V každém modulu má uživatel jiné možnosti, co dělat s ticketem. Běžný uživatel má pouze možnost vytvořit ticket, vybrat servery, skupiny práva, napsat komentář a uložit ticket.

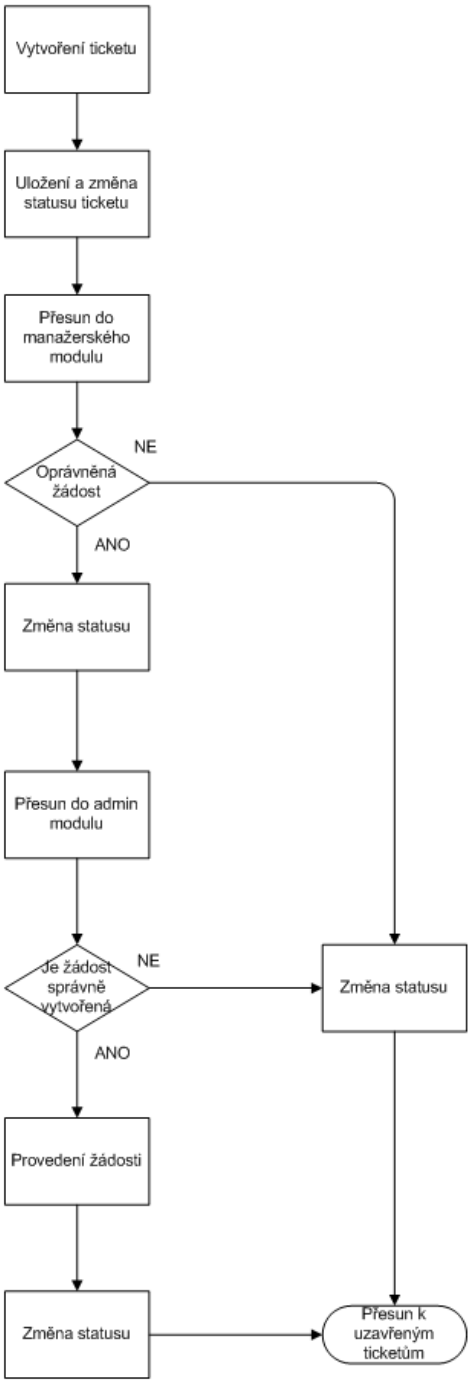
V manažerském módu se zobrazí fronta ticketů, které čekají na schválení jejich nadřízeným. Pro něj to bude otázka zkontrolování, kam si který uživatel žádá přístup a buď změní status na schváleno nebo na zamítnuto. V případě schválení se ticket přesune do administrátorského modulu, v případě zamítnutí se ticket uloží do databáze a již se nikde nezobrazuje, avšak je možné jej dohledat. Do komentáře může kdokoliv (kdo má s ticketem co do činění) napsat jakoukoliv poznámku, například důvod, proč chce být vytvořen nebo důvod, proč byl zamítnut.

Administrátorský mód zobrazí frontu schválených ticketů a pokud ticket obsahuje všechny náležitosti (například pokud si uživatel vytvoří ticket na reset hesla, tak uživatel

musí být již vytvořený - interní pravidla), administrátor ho provede tak, jak žádal v ticketu a změní status ticketu na dokončen.

5.3 PROCES TICKETU

Zde vidíte diagram, který popisuje, co se děje od vytvoření ticketu s požadavkem až po jeho dokončení administrátory.



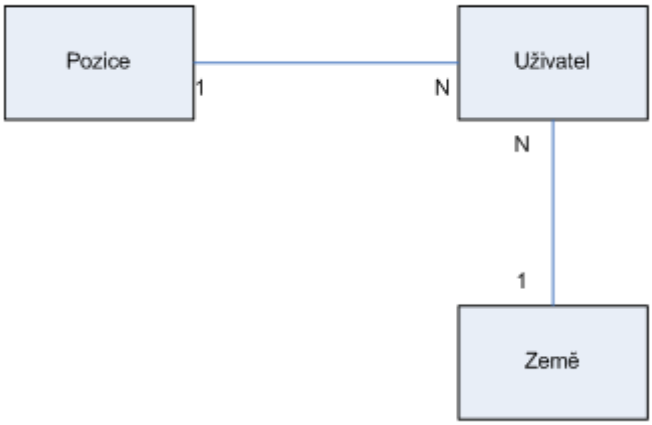
Obrázek 12:*Proces cesty ticketu*

5.4 DEKOMPOZICE VZTAHŮ V DATABÁZI

Základní vztah v databázi je takový, že uživatel vytváří tickety. Jeden uživatel může vytvořit neomezené množství ticketů.

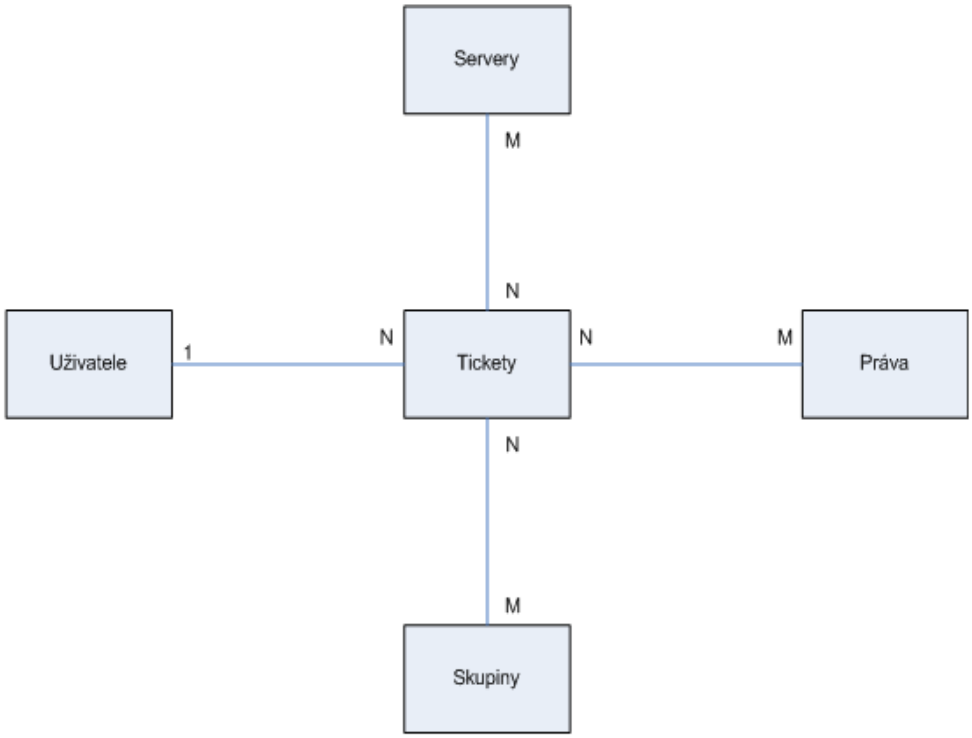
Obrázek 13:*Vztah uživatel-ticekt*

Více uživatelů má některé společné vlastnosti, jako zemi, ze které pochází a například pozici, na které se (pro systém) nachází. Proto jsou vytvořeny tabulky pozic a zemí, v nichž jsou uloženy tyto společné informace.



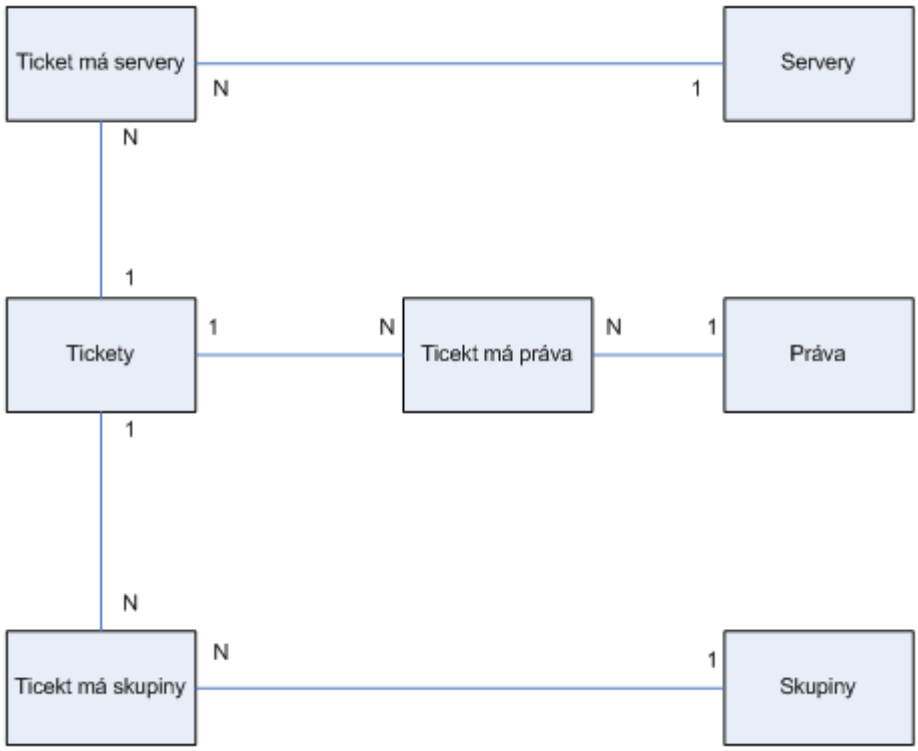
Obrázek 14: *Vztah uživatel-země, uživatel-pozice*

V ticketech si uživatel vybírá, jaké bude chtít vytvořit servery, jaké bude chtít mít skupiny a jaká bude potřebovat práva. Jeden uživatel může vytvořit více ticketů a v každém si může zažádat o libovolný počet serverů, skupin či práv.



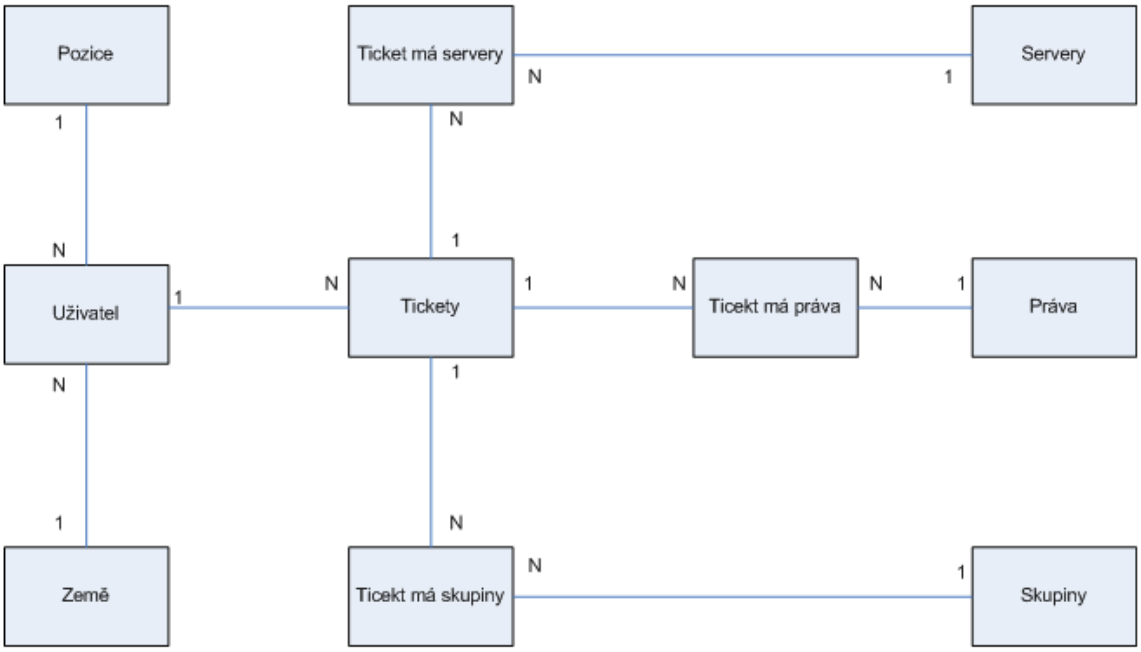
Obrázek 15: *Vztahy ticketu s okolím*

Aby více ticketů mohlo obsahovat více serverů, práv a skupin, je třeba provést dekompozici těchto vztahů:



Obrázek 16: *Dekompozice vztahů ticeketu s okolím*

po složení jednotlivých vztahů dostaneme konceptuální schéma databáze, které tedy vypadá takto:



Obrázek 17:*Konceptuální schéma databáze*

5.5 JEDNOTLIVÉ TABULKY DATABÁZE

Tabulka uživatelů	
uzivatel_id	INT
jmeno	varchar (200)
uzivatelske_jme	varchar (20)
heslo	varchar (50)
email	varchar (100)
staty_id	INT
pozice_id	INT

Tabulka 1: *Uživatelé*

Tabulka uživatelů je jednou z nejdůležitějších tabulek v databázi. Z této tabulky informační systém bere data pro přihlašování uživatelů a generuje z nich potřebné informace pro administrátory. Primárním klíčem tabulky je uživatel_id, který je svázán s tabulkou ticket. První cizí klíč je staty_id, jenž je svázán s tabulkou států. Druhý cizí klíč je pozice_id, která je svázaná s tabulkou pozic.

Tabulka ticketů	
ticket_id	INT
uzivatel_id	INT
vytvoren	datetime
status	INT
comment	varchar(400)

Tabulka 2: *Tickety*

Tabulka ticketů je nejdůležitější tabulkou v databázi. Je na ni nejvíce vazeb z ostatních tabulek a řeší evidenci všech vytvořených požadavků od uživatelů na adminy. Dále má pro informační systém vysoký význam položka status, jež určuje, v jakém modulu se ticket bude zobrazovat. Primárním klíčem je ticket_id, který je svázán na tabulky práv, skupin, serverů. Cizím klíčem je zde uživatel_id, který svazuje tabulku ticketů s tabulkou uživatelů.

Tabulka států	
stat ID	INT
kod_statu	varchar (3)
nazev	varchar (100)

Tabulka 3:*Státy*

Tabulka pozic	
pozice_id	INT
nazev_pozice	varchar(100)

Tabulka 4: *Pozice*

Tabulka států řeší, z jaké země je uživatel, neboť pro adminy je to důležitá informace, již musí zohlednit dle interních pravidel při vytváření uživatelských účtů. Primárním klíčem je stat_id, který je svázán s cizím klíčem staty_id v tabulce uživatelů.

Tabulka pozic identifikuje uživatele v rámci informačního systému. Dle hodnot v ní uložených informační systém pozná, který modul má uživateli zobrazit. Primárním klíčem tabulky pozic e pozice_id. Tento primární klíč je svázán s cizím klíčem pozice_id v tabulce uživatelů.

Tabulka dekompozice 1	
ticket_id	INT
serever_id	INT

Tabulka 5: *Tabulka dekompozice*

Tabulka dekompozice 2	
ticket_id	INT
skupina_id	INT

Tabulka 6: *Tabulka dekompozice*

Tabulka dekompozice 3	
ticket_id	INT
prava_id	INT

Tabulka 7: *Tabulka dekompozice*

Tyto tabulky byly vytvořeny za účelem dekompozice vztahů M:N. Primární klíče jsou tvořeny server_id, skupina_id, prava_id. Cizí klíč je ve všech třech tabulkách ticket_id. JE provázán s tabulkou ticketů.

Servery	
serever_id	INT
nazev	varchar(100)
ip_adresa	varchar(50)

Tabulka 8:*Servery*

Skupiny	
skupina_id	INT
nazev	varchar(100)

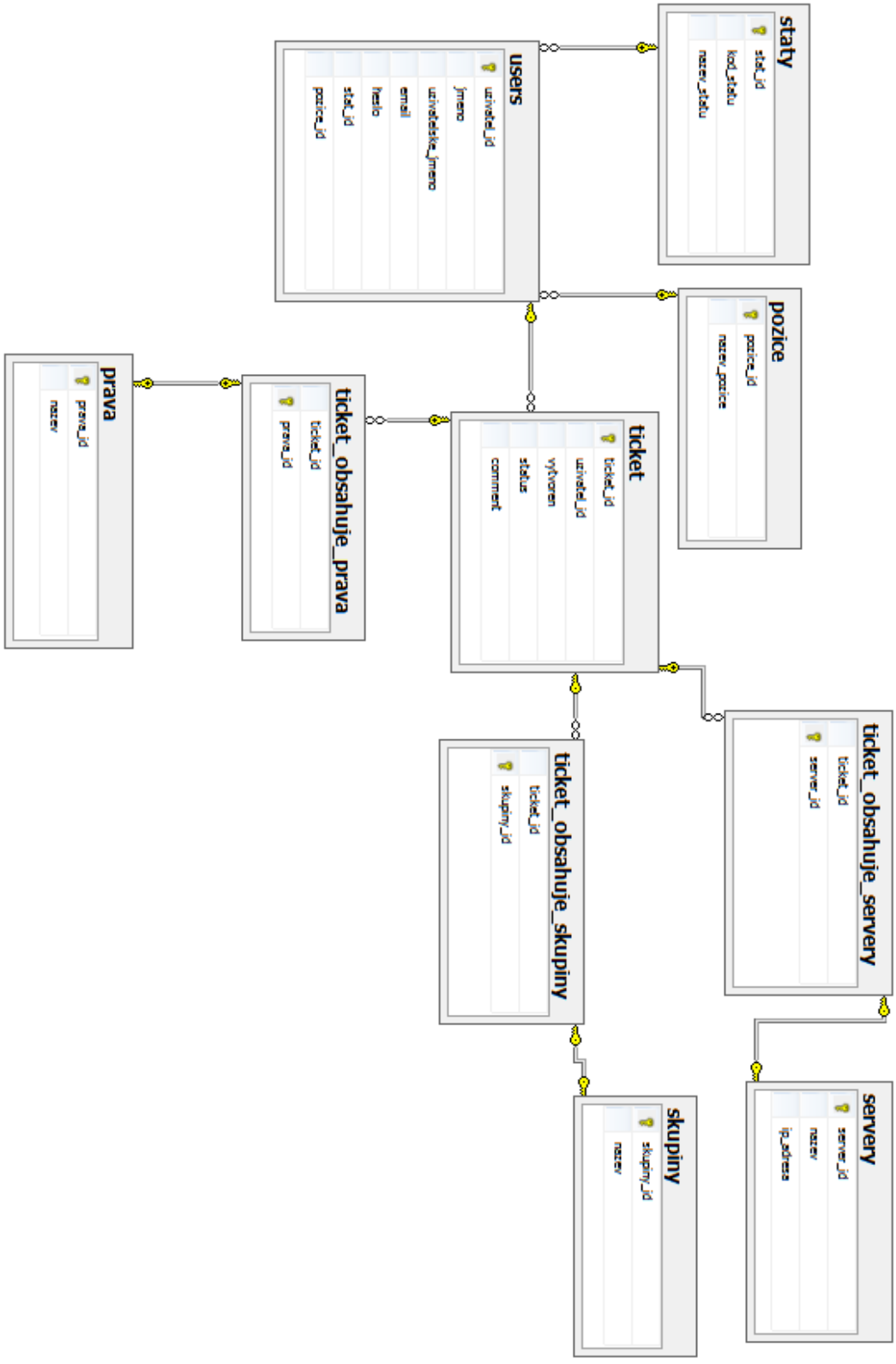
Tabulka 9:*Skupiny*

Práva	
prava_id	INT
nazev	varchar(100)

Tabulka 10:*Práva*

Poslední tři tabulky jsou ve formě číselníků a ukládají informace o serverech včetně ip adresy, skupinách, které mají uživatelé přidány, a o právech. Primání klíče jsou opět tvořeny server_id, skupina_id, prava_id a jsou svázány se stejnými klíči v dekompozičních tabulkách.

5.6 ER DIAGRAM



Obrázek 18: *ER diagram vytvořené databáze*

5.7 FYZICKÁ TVORBA DATABÁZE

```
USE [master]
GO

CREATE DATABASE [IBM_user accounts] ON  PRIMARY
(   NAME   =   N'IBM_user   accounts',   FILENAME   =
N'C:\DATABASE\IBM_user accounts.mdf' , SIZE = 2048KB , MAX-
SIZE = UNLIMITED, FILEGROWTH = 1024KB )
 LOG ON
(   NAME   =   N'IBM_user   accounts_log',   FILENAME   =
N'C:\DATABASE\IBM_user accounts_log.ldf' , SIZE = 1024KB ,
MAXSIZE = 2048GB , FILEGROWTH = 10%)
GO

ALTER  DATABASE  [IBM_user  accounts]  SET  COMPATIBILI-
TY_LEVEL = 100
GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC  [IBM_user  accounts].[dbo].[sp_fulltext_database]
@action = 'enable'
end
GO



USE [IBM_user accounts]
GO

CREATE TABLE [dbo].[users](
     [uzivatel_id] [int] NOT NULL,
     [jmeno] [varchar](200) NOT NULL,
     [uzivatelske_jmeno] [varchar](20) NOT NULL,
     [email] [varchar](200) NOT NULL,
     [heslo] [varchar](50) NOT NULL,
     [stat_id] [int] NOT NULL,
     [pozice_id] [int] NOT NULL,
 CONSTRAINT [PK_users_1] PRIMARY KEY CLUSTERED
(
     [uzivatel_id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY   =   OFF,   ALLOW_ROW_LOCKS      =   ON,   AL-
LOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]

GO
```

```
     ALTER TABLE [dbo].[users]  WITH CHECK ADD  CONSTRAINT
[FK_users_pozice] FOREIGN KEY([pozice_id])
     REFERENCES [dbo].[pozice] ([pozice_id])
     GO

     ALTER    TABLE    [dbo].[users]    CHECK    CONSTRAINT
[FK_users_pozice]
     GO

     ALTER TABLE [dbo].[users]  WITH CHECK ADD  CONSTRAINT
[FK_users_staty] FOREIGN KEY([stat_id])
     REFERENCES [dbo].[staty] ([stat_id])

     GO

     ALTER    TABLE    [dbo].[users]    CHECK    CONSTRAINT
[FK_users_staty]
     GO


     CREATE TABLE [dbo].[ticket](
          [ticket_id] [int] NOT NULL,
          [uzivatel_id] [int] NOT NULL,
          [vytvoren] [datetime] NOT NULL,
          [status] [int] NOT NULL,
          [comment] [varchar](400) NULL,
      CONSTRAINT [PK_ticket] PRIMARY KEY CLUSTERED
     (
          [ticket_id] ASC
     )WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY   =   OFF,   ALLOW_ROW_LOCKS      =   ON,   AL-
LOW_PAGE_LOCKS  = ON) ON [PRIMARY]
     ) ON [PRIMARY]

     GO

     ALTER TABLE [dbo].[ticket]  WITH CHECK ADD  CONSTRAINT
[FK_ticket_users] FOREIGN KEY([uzivatel_id])
     REFERENCES [dbo].[users] ([uzivatel_id])
     GO

     ALTER    TABLE    [dbo].[ticket]    CHECK    CONSTRAINT
[FK_ticket_users]
     GO



     CREATE TABLE [dbo].[staty](
          [stat_id] [int] NOT NULL,
```

```
          [kod_statu] [varchar](3) NOT NULL,
          [nazev_statu] [varchar](100) NOT NULL,
      CONSTRAINT [PK_staty] PRIMARY KEY CLUSTERED
     (
          [stat_id] ASC
     )WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY   =   OFF,   ALLOW_ROW_LOCKS      =   ON,   AL-
LOW_PAGE_LOCKS  = ON) ON [PRIMARY]
     ) ON [PRIMARY]



     CREATE TABLE [dbo].[ticket_obsahuje_prava](
          [ticket_id] [int] NOT NULL,
          [prava_id] [int] NOT NULL,
      CONSTRAINT   [PK_ticket_obsahuje_prava]   PRIMARY   KEY
CLUSTERED
     (
          [prava_id] ASC
     )WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY   =   OFF,   ALLOW_ROW_LOCKS      =   ON,   AL-
LOW_PAGE_LOCKS  = ON) ON [PRIMARY]
     ) ON [PRIMARY]

     GO

     ALTER  TABLE  [dbo].[ticket_obsahuje_prava]   WITH  CHECK
ADD    CONSTRAINT  [FK_ticket_obsahuje_prava_prava]  FOREIGN
KEY([prava_id])
     REFERENCES [dbo].[prava] ([prava_id])
     GO

     ALTER  TABLE  [dbo].[ticket_obsahuje_prava]  CHECK  CON-
STRAINT [FK_ticket_obsahuje_prava_prava]
     GO

     ALTER  TABLE  [dbo].[ticket_obsahuje_prava]   WITH  CHECK
ADD    CONSTRAINT  [FK_ticket_obsahuje_prava_ticket]  FOREIGN
KEY([ticket_id])
     REFERENCES [dbo].[ticket] ([ticket_id])
     GO

     ALTER  TABLE  [dbo].[ticket_obsahuje_prava]  CHECK  CON-
STRAINT [FK_ticket_obsahuje_prava_ticket]
     GO



     CREATE TABLE [dbo].[ticket_obsahuje_servery](
          [ticket_id] [int] NOT NULL,
```

```
          [server_id] [int] NOT NULL,
      CONSTRAINT  [PK_ticket_obsahuje_servery]  PRIMARY  KEY
CLUSTERED
     (
          [server_id] ASC
     )WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY   =   OFF,   ALLOW_ROW_LOCKS      =   ON,   AL-
LOW_PAGE_LOCKS  = ON) ON [PRIMARY]
     ) ON [PRIMARY]

     GO

     ALTER TABLE [dbo].[ticket_obsahuje_servery]  WITH CHECK
ADD  CONSTRAINT [FK_ticket_obsahuje_servery_ticket] FOREIGN
KEY([ticket_id])
     REFERENCES [dbo].[ticket] ([ticket_id])
     GO

     ALTER TABLE [dbo].[ticket_obsahuje_servery] CHECK CON-
STRAINT [FK_ticket_obsahuje_servery_ticket]
     GO



     CREATE TABLE [dbo].[ticket_obsahuje_skupiny](
          [ticket_id] [int] NOT NULL,
          [skupiny_id] [int] NOT NULL,
      CONSTRAINT  [PK_ticket_obsahuje_skupiny]  PRIMARY  KEY
CLUSTERED
     (
          [skupiny_id] ASC
     )WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY   =   OFF,   ALLOW_ROW_LOCKS      =   ON,   AL-
LOW_PAGE_LOCKS  = ON) ON [PRIMARY]
     ) ON [PRIMARY]

     GO

     ALTER TABLE [dbo].[ticket_obsahuje_skupiny]  WITH CHECK
ADD  CONSTRAINT [FK_ticket_obsahuje_skupiny_skupiny] FOREIGN
KEY([skupiny_id])
     REFERENCES [dbo].[skupiny] ([skupiny_id])
     GO

     ALTER TABLE [dbo].[ticket_obsahuje_skupiny] CHECK CON-
STRAINT [FK_ticket_obsahuje_skupiny_skupiny]
     GO
```

```
     ALTER TABLE [dbo].[ticket_obsahuje_skupiny]  WITH CHECK
ADD   CONSTRAINT [FK_ticket_obsahuje_skupiny_ticket] FOREIGN
KEY([ticket_id])
     REFERENCES [dbo].[ticket] ([ticket_id])
     GO

     ALTER TABLE [dbo].[ticket_obsahuje_skupiny] CHECK CON-
STRAINT [FK_ticket_obsahuje_skupiny_ticket]
     GO



     CREATE TABLE [dbo].[pozice](
          [pozice_id] [int] NOT NULL,
          [nazev_pozice] [varchar](100) NOT NULL,
      CONSTRAINT [PK_pozice] PRIMARY KEY CLUSTERED
     (
          [pozice_id] ASC
     )WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY   =   OFF,   ALLOW_ROW_LOCKS      =   ON,   AL-
LOW_PAGE_LOCKS  = ON) ON [PRIMARY]
     ) ON [PRIMARY]



     CREATE TABLE [dbo].[prava](
          [prava_id] [int] NOT NULL,
          [nazev] [varchar](100) NOT NULL,
      CONSTRAINT [PK_prava] PRIMARY KEY CLUSTERED
     (
          [prava_id] ASC
     )WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY   =   OFF,   ALLOW_ROW_LOCKS      =   ON,   AL-
LOW_PAGE_LOCKS  = ON) ON [PRIMARY]
     ) ON [PRIMARY]



     CREATE TABLE [dbo].[servery](
          [server_id] [int] NOT NULL,
          [nazev] [varchar](100) NOT NULL,
          [ip_adresa] [varchar](50) NULL,
      CONSTRAINT [PK_servery] PRIMARY KEY CLUSTERED
     (
          [server_id] ASC
     )WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY   =   OFF,   ALLOW_ROW_LOCKS      =   ON,   AL-
LOW_PAGE_LOCKS  = ON) ON [PRIMARY]
     ) ON [PRIMARY]
```

```
     GO

     CREATE TABLE [dbo].[skupiny](
          [skupiny_id] [int] NOT NULL,
          [nazev] [varchar](100) NOT NULL,
      CONSTRAINT [PK_skupiny] PRIMARY KEY CLUSTERED
     (
          [skupiny_id] ASC
     )WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
IGNORE_DUP_KEY   =   OFF,   ALLOW_ROW_LOCKS      =   ON,   AL-
LOW_PAGE_LOCKS  = ON) ON [PRIMARY]
     ) ON [PRIMARY]
```

6. ZÁVĚR

Ve této práci jsem řešil problémy, se kterými se denně setkávám. Tyto problémy zpomalují celý pracovní proces, dochází kvůli nim ke zbytečným konfliktům a hlavně jsou v podstatě nesmyslné. Pro řízení uživatelských účtů zde nebyl vytvořen informační systém, vše se řešilo formou obyčejného požadavku, například emailem. Sestavil jsem tedy databázi, která eviduje veškeré důležité informace o uživatelích, jejich účtech a jejich právech. Samotná databáze přinesla jen nám administrátorům přehled a umožnila nám rychleji a pružněji reagovat na zákaznické požadavky. Přestali se řešit problémy, kdy nebylo jasné, zda uživatel má či nemá na serveru účet a jestli byla jeho žádost zamítnuta oprávněně či nikoliv. Databáze byla navržena pružně s ohledem na budoucí automatizovaný informační systém. Ten je v současné době ve vývoji a v testovací fázi, ale vytvořená databáze obsahovala téměř vše potřebné, takže ani není důležité provádět nějaké zásadní změny. Informační systém zautomatizoval a zjednodušil vytváření požadavků a výrazně zjednodušil proces schvalování ticketů. Databáze je jednodušší, než jsem předpokládal, avšak přináší velká pozitiva do evidence uživatelů a do bezpečnosti systémů, neboť neoprávněné osoby již nemají možnost získat přístup k serverům s citlivými aplikacemi. Ve vysokém množství účtů, uživatelů a serverů tato jednoduchá databáze pomohla snížit množství osob v týmu pracujících na uživatelských účtech z pěti na tři, což je z dlouhodobého hlediska výrazný ekonomický přínos. Pokud bude ze strany firmy rozšířit informační systém a databázi i do ostatních týmů, případně poboček a zemí, může být tato databáze velice přínosná.

LITERÁRNÍ ZDROJE

1) BASL, J. – BLAŽÍČEK, R. *Podnikové informační systémy*. 2.vyd. Praha: Grada Publishing, 2008. 288s. ISBN 978-80-247-2279-5

2) KOCH, M. *Datové a funkční modelování.2.vyd.* Brno: CERM,2006. 108s. ISBN 80-214-3252-7

3) GROFF, James R., WEINBERG, Paul N. *SQL Kompletní průvodce.* Brno : CP Books, a.s., 2005. 936 s., CD-ROM. ISBN 80-251-0369-2

4) DOSEDĚL, T. *Počítačová bezpečnost a ochrana dat*. 1. vydání. Brno: Computer Press, 2004. 190 s. ISBN 80-251-0106-1

5) HOTEK, M. *Microsoft SQL Server 2008: krok za krokem*. 1.vydání. Brno: Computer Press, 2009. 488 s. ISBN 978-80-251-2466-6

6) KOTLER, P. *Marketing management*. 1. vydání. Praha: Grada, 2007. 788 s. ISBN 978-80-247-1359-5

7) PUŽMANOVÁ, R. *Moderní komunikační sítě od A do Z*. 2. aktualizované vydání. Brno: Computer Press, 2006. 430 s. ISBN 80-251-1278-0

8) ŘEPA, V. *Podnikové procesy. 2.rozšířené vydání*. Praha: Grada, 2007. 281 s. ISBN 978-80-247-2252-8

9) VOŘÍŠEK, J. *Principy a modely řízení podnikové informatiky*. 1. vydání. Praha: Oeconomica, 2008. 446 s. ISBN 978-80-245-1440-6

WEBOVÉ ZDROJE

1) MENČÍK, Pavel. Úvod do informačních systémů. [online]. [cit. 2010-03-05]. Dostupné z: < http://pmencik.sweb.cz/zvi.htm/>

2) Microsoft. Microsoft SQL 2008. [online]. [cit. 2010-03-05]. Dostupné z: <http://www.microsoft.com/cze/sqlserver2008/overview.mspx>

3) WIKI,.HyperText markup Language [online]. [cit. 2010-03-05]. Dostupné z: <http://http://cs.wikipedia.org/wiki/HyperText_Markup_Language>

4) PHP Group,.Historie php. [online]. [cit. 2010-03-05]. Dostupné z: http://php.tonnikala.org/manual/cs/history.php

5)) ZELENKA, Petr. WebML – datové modelování. [online]. [cit. 2010-03-05]. Dostupné z: < http://interval.cz/clanky/webml-datove-modelovani/>

SEZNAM OBRÁZKŮ:

SEZNAM TABULEK: